기타

최백준 choi@startlink.io

시험 감독

https://www.acmicpc.net/problem/13458

- N개의 시험장이 있다
- i번 시험장에 있는 사람의 수는 A[i]명
- 총감독관은 한 방에서 감시할 수 있는 응시자의 수가 B명
- 부감독관은 한 방에서 감시할 수 있는 응시자의 수가 C명
- 각각의 시험장에 총감독관은 1명, 부감독관은 여러명이 가능
- 모든 응시자를 감시해야 했을 때, 필요한 감독관 수의 최소값

시험 감독

https://www.acmicpc.net/problem/13458

- 각각의 방마다 한 명의 총감독관이 있다고 가정하고
- 나머지 인원을 감시하는데 필요한 부감독관의 수를 구한다

시험 감독

https://www.acmicpc.net/problem/13458

- 각각의 방마다 한 명의 총감독관이 있다고 가정하고
- 나머지 인원을 감시하는데 필요한 부감독관의 수를 구한다
- i번 방에 필요한 총감독관의 수 = 1
- 감시가 필요한 사람의 수 = A[i] – B
  - 이 값이 0보다 크면, i번 방에 필요한 부감독관의 수 = (A[i] – B) / C + ((A[i]-B) % C == 0 ? 0 : 1)

시험 감독

https://www.acmicpc.net/problem/13458

- 소스: http://codeplus.codes/6ed75dbbe3a24546967ef716e5ab9c15

AB

https://www.acmicpc.net/problem/12970

- 정수 N과 K가 주어졌을 때, 다음 두 조건을 만족하는 문자열 S를 찾는 문제
- 문자열 S의 길이는 N이고, 'A', 'B'로 이루어져 있다.
- 문자열 S에는 $0 \leq i < j < N$ 이면서 s[i] == 'A' && s[j] == 'B'를 만족하는 (i, j) 쌍이 K개가 있다.

AB

https://www.acmicpc.net/problem/12970

- A가 a개, B가 b개가 있으면
- (i, j)쌍이 0 ~ a*b가 되는 문자열을 항상 만들 수 있다

AB

https://www.acmicpc.net/problem/12970

- 먼저, B를 b개를 놓고,
- a를 어디에 추가하면 좋을지 선택한다.

AB

https://www.acmicpc.net/problem/12970

- 소스: http://codeplus.codes/d3771f46aaa94e88abb476ed13463e45

a^b

a^b

- a의 b제곱을 빠르게 구해야 한다.

```
int ans = 1;
for (int i=1; i<=b; i++) {
    ans = ans * a;
}
```

- 직관적인 방법이지만 O(b)라는 시간이 걸리게 된다.
- 따라서, 조금 더 빠른 방법이 필요하다.

a^b

분할정복 이용하기

- 분할정복을 이용해서 구할 수 있다.
- $a^{2b} = a^b \times a^b$
- $a^{2b+1} = a \times a^{2b}$

a^b

분할정복 이용하기

```
int calc(int a, int b) {
    if (b == 0) {
        return 1;
    } else if (b == 1) {
        return a;
    } else if (b % 2 == 0) {
        int temp = calc(a, b/2);
        return temp * temp;
    } else { // b % 2 == 1
        return a * calc(a, b-1);
    }
}
```

a^b

분할정복 이용하기

- 이 부분을

```
} else if (b % 2 == 0) {
    int temp = calc(a, b/2);
    return temp * temp;
}
```

- 아래와 같이 구현 하면 O(N) 이다. (호출이 2배가 되어버린다)

```
} else if (b % 2 == 0) {
    return calc(a, b/2) * calc(a, b/2);
}
```

a^b

분할정복 이용하기

- 이진수의 원리를 이용해서도 구할 수 있다.

```
int calc(int a, int b) {
    int ans = 1;
    while (b > 0) {
        if (b % 2 == 1) {
            ans *= a;
        }
        a = a * a;
        b /= 2;
    }
    return ans;
}
```

a^b

분할정복 이용하기

- 예를 들어, 3의 27 제곱인 경우를 생각해보자.
- 27은 이진수로 11011 이다.
- $27 = 2^0 + 2^1 + 2^3 + 2^4$
- $27 = 1 + 2 + 8 + 16$
- $3^{27} = 3^{1+2+8+16}$
- $3^{27} = 3^1 \times 3^2 \times 3^8 \times 3^{16}$
- 을 이용해서 a를 계속해서 a*a로 곱해가면서 제곱을 구하게 된다.

곱셈

https://www.acmicpc.net/problem/1629

- 자연수 A를 B번 곱한 수를 C로 나눈 나머지를 구하는 문제

곱셈

https://www.acmicpc.net/problem/1629

- 분할 정복: http://codeplus.codes/01badd051f4948548df26f06028984a7
- 이진수 응용: http://codeplus.codes/50ff2fc9db824fe68936a293e928d4fc

파스칼의 삼각형

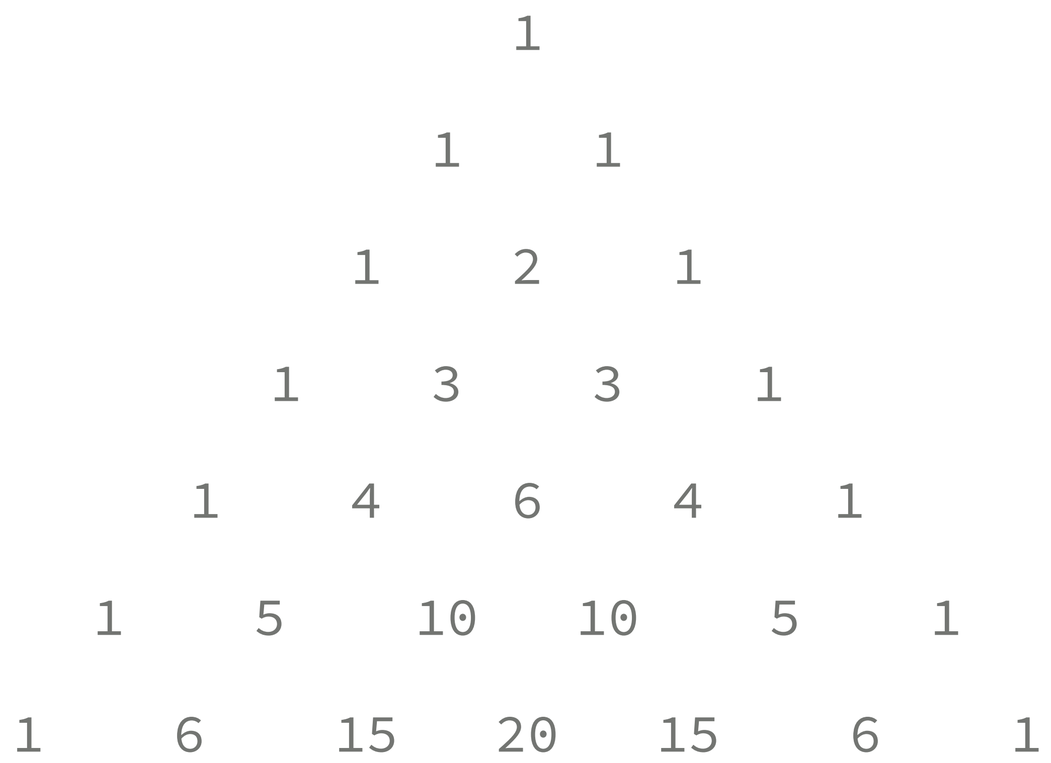
Pascal's Triangle

- 이항 계수를 삼각형 모양으로 배열
- n번 줄에는 수를 n개만 쓴다.
- 각 줄의 첫 번째와 마지막 수는 1이다.
- 나머지 수는 윗 줄의 왼쪽 수와 오른쪽 수를 더해서 만든다.

파스칼의 삼각형

Pascal's Triangle

- 5까지 파스칼의 삼각형



파스칼의 삼각형

Pascal's Triangle

- C[n][k] = n번줄의 k번째 수라고 했을 때
- C[n][1] = 1, C[n][n] = 1
- C[n][k] = C[n-1][k-1] + C[n-1][k]
- 로 정의할 수 있음
- C[n][k]는 $\binom{n}{k}$ 이다.

파스칼의 삼각형

Pascal's Triangle

- C[n][k] = C[n-1][k-1] + C[n-1][k]
- C[n][k] 는 $\binom{n}{k}$ 를 나타내기 때문에, n개 중에 k개를 순서 없이 고르는 방법이다.
- n개 중에 k개를 순서 없이 고른다면 다음과 같은 두 가지 경우가 가능하다

1. n번째를 고른 경우
2. n번째를 고르지 않은 경우

파스칼의 삼각형

Pascal's Triangle

- C[n][k] = C[n-1][k-1] + C[n-1][k]
- C[n][k] 는 $\binom{n}{k}$ 를 나타내기 때문에, n개 중에 k개를 순서 없이 고르는 방법이다.
- n개 중에 k개를 순서 없이 고른다면 다음과 같은 두 가지 경우가 가능하다

1. n번째를 고른 경우
    - n번째를 골랐기 때문에, n-1개 중에 k-1개를 골랐어야 한다. C[n-1][k-1]
2. n번째를 고르지 않은 경우
    - n번째를 고르지 않았기 때문에, n-1개 중에 k개를 골랐어야 한다. C[n-1][k]

이항 계수 2

https://www.acmicpc.net/problem/11051

- $\binom{n}{k}$ 을 10,007로 나눈 나머지를 구하는 문제
- $1 \leq n \leq 1{,}000, 0 \leq k \leq n$ 이기 때문에
- 파스칼의 삼각형을 이용해서 구하면 된다.

이항 계수 2

https://www.acmicpc.net/problem/11051

- 소스: http://codeplus.codes/7373685e614f4248b09c4370396fb117

누적합

Prefix Sum

- 수열 A[1], A[2], …, A[N]이 있을 때
- A[i] + … + A[j]를 구하는 문제

- S[i] = A[1] + A[2] + … + A[i]

누적합

Prefix Sum

- A[i] + … + A[j]를 구하는 문제

- S[j] = A[1] + A[2] + … + A[i-1] + A[i] + … + A[j]
- S[i-1] = A[1] + A[2] + … + A[i-1]
- S[j] – S[i-1] = A[i] + … + A[j]

구간 합 구하기 4

https://www.acmicpc.net/problem/11659

- 수 N개가 주어졌을 때, i번째 수부터 j번째 수까지 합을 구하는 문제

구간 합 구하기 4

https://www.acmicpc.net/problem/11659

- 소스: http://codeplus.codes/5031c62b0bb246dea18c461705f16094

나머지 합

https://www.acmicpc.net/problem/10986

- 수 N개 A[1], A[2], ..., A[N]이 주어진다.
- 연속된 부분 구간의 합이 M으로 나누어 떨어지는 구간의 개수를 구하는 문제
- 즉, A[i] + ... + A[j] (i ≤ j) 의 합이 M으로 나누어 떨어지는 (i, j) 쌍의 개수를 구해야 한다.

나머지 합

https://www.acmicpc.net/problem/10986

- S[i] = A[1] + … + A[i] 라고 하자
- A[i] + … + A[j] = S[j] – S[i-1]
- (A[i] + … + A[j]) % M = (S[j] – S[i-1]) % M

- (A[i] + … + A[j]) % M == 0 인 것의 개수를 구해야 한다
- (S[j] – S[i-1]) % M == 0 와 같다
- 나눈 나머지가 0이 되려면
- S[j] % M == S[i-1] % M 이 되어야 한다

나머지 합

https://www.acmicpc.net/problem/10986

- 이 문제는
- S[j] % M == S[i-1] % M 이 되어야 한다
- 를 만족하는 (i, j) 쌍의 개수를 구하는 문제가 된다.
- cnt[k]를 S[i] % M == k 인 i의 개수라고 하면
- $0 \leq k < M$인 k에 대해서
- cnt[k] * (cnt[k] - 1) / 2 의 합을 구하면 된다.

나머지 합

https://www.acmicpc.net/problem/10986

- 소스: http://codeplus.codes/6095fc15e71944c18007c703ae3b9057

책 페이지

https://www.acmicpc.net/problem/1019

- 총 페이지 수가 N인 책이 있다
- 첫 페이지 1
- 마지막 페이지 N
- 0부터 9까지 각 숫자가 몇 번씩 나오는지 구하는 문제
- $N \leq 1,000,000,000$

책 페이지

https://www.acmicpc.net/problem/1019

- 11의 경우
- 1 2 3 4 5 6 7 8 9 10 11
- 0: 1개
- 1: 4개
- 2, 3, 4, 5, 6, 7, 8, 9: 1개

- 정답
- 1 4 1 1 1 1 1 1 1 1

책 페이지

https://www.acmicpc.net/problem/1019

- 1부터 N까지 모든 페이지에 대해서, 각 숫자가 몇 번씩 나오는지 세보는 방법
- 시간 복잡도: O(N)
- $N \leq 1{,}000{,}000{,}000$
- 보통 1억번 정도가 1초인데, 10억이기 때문에 시간초과를 받게 된다

책 페이지

https://www.acmicpc.net/problem/1019

- 1부터 N까지가 아닌 A부터 B까지 각 숫자가 모두 몇 번 나오는지 구하는 문제로 변형해서 해결해보자
- A는 일의 자리가 0이 되어야 하고, B는 일의 자리가 9가 되어야 한다
- 아니면, A++ 또는 B-- 를 통해서 맞추어 준다.

책 페이지

https://www.acmicpc.net/problem/1019

- A = 1345
- B = 8742

- 일단 A의 일의 자리가 0이 아니고, B의 일의 자리가 9가 아니다.
- 이 때, A++, B--를 하면서 각 수에 들어있는 자리수를 계산한다.
- 이렇게 되면 A = 1350, B = 8739로 변한다.

책 페이지

https://www.acmicpc.net/problem/1019

- A = 1350
- B = 8739

- A부터 B까지 일의 자리에 0~9는 총 (873-135+1)번 등장한다

책 페이지

https://www.acmicpc.net/problem/1019

- A = 10
- B = 39
- 인 경우에

- **10**, 1**1**, 1**2**, 1**3**, 1**4**, 1**5**, 1**6**, 1**7**, 1**8**, 1**9**
- **20**, 2**1**, 2**2**, 2**3**, 2**4**, 2**5**, 2**6**, 2**7**, 2**8**, 2**9**
- **30**, 3**1**, 3**2**, 3**3**, 3**4**, 3**5**, 3**6**, 3**7**, 3**8**, 3**9**

책 페이지

https://www.acmicpc.net/problem/1019

- A = 1350
- B = 8739

- A부터 B까지 일의 자리에 0~9는 총 (873-135+1)번 등장한다
- 이렇게 해서 ans[0]~ans[9]에 (873-135+1)를 더할 수 있다

- 이제 일의 자리는 모두 계산이 끝났기 때문에, 십의 자리를 계산해야 한다
- A = A/10, B = B/10을 하자

책 페이지

https://www.acmicpc.net/problem/1019

- A = 135
- B = 873

- A가 0으로, B가 9로 끝나지 않기 때문에, 일단 두 수를 ++, --시킨다.
- 135 -> 136이 될 때
- 사실은 135? -> 136? 이 되는 것이다.
- 즉, 136이 되려면 1이 10번, 3이 10번, 5가 10번 등장하는 것이다

책 페이지

https://www.acmicpc.net/problem/1019

- A = 140
- B = 869

- 그럼 이제, 0~9는 총 (86-14+1) 번 등장한다?
- 아니다
- 여기는 십의 자리이기 때문에, 총 (86-14+1)*10 번 등장한다

책 페이지

https://www.acmicpc.net/problem/1019

- A = 10
- B = 39
- 인 경우에

- **10**, 1**1**, 1**2**, 1**3**, 1**4**, 1**5**, 1**6**, 1**7**, 1**8**, 1**9**
- **20**, 2**1**, 2**2**, 2**3**, 2**4**, 2**5**, 2**6**, 2**7**, 2**8**, 2**9**
- **30**, 3**1**, 3**2**, 3**3**, 3**4**, 3**5**, 3**6**, 3**7**, 3**8**, 3**9**
- 저렇게 저 숫자들 뒤에 전부 0~9까지가 붙은 수가 등장하는 것
- 즉, **10**은 사실 **100**, **101**, **102**, …, **109**가 하나로 뭉쳐져 있는 것

책 페이지

https://www.acmicpc.net/problem/1019

- A = 140
- B = 869

- 그럼 이제, 0~9는 총 (86-14+1) 번 등장한다?
- 아니다
- 여기는 십의 자리이기 때문에, 총 (86-14+1)*10 번 등장한다
- A = A/10, B = B/10을 하고 백의 자리를 살펴보자

책 페이지

https://www.acmicpc.net/problem/1019

- 이런식으로 계속 진행한다.

책 페이지

https://www.acmicpc.net/problem/1019

- 소스: http://codeplus.codes/e342717e70914acb94160df3fa5779ce

무한 문자열

https://www.acmicpc.net/problem/12871

- 문자열 s가 있을 때, f(s)는 s를 무한번 붙인 문자열
- s = "abc" 인 경우에 f(s) = "abcabcabcabc..."
- f(s)와 f(t)가 같은 문자열을 만드는지 아닌지 구하는 프로그램을 작성하시오.
- s와 t의 길이 $\leq$ 50

무한 문자열

https://www.acmicpc.net/problem/12871

- 문자열을 길이가 같아질 때까지 만들어보면 된다.
- 두 문자열 길이의 제한이 50이기 때문에, 50*50 = 2500 까지 검사해보면 된다.

무한 문자열

https://www.acmicpc.net/problem/12871

```
string s, t;
cin >> s >> t;
for (int i=0; i<2500; i++) {
    if (s[i%s.length()] != t[i%t.length()]) {
        cout << 0 << '\n';
        return 0;
    }
}
cout << 1 << '\n';
```

무한 문자열

https://www.acmicpc.net/problem/12871

```
string S = s, T = t;
while (S.length() != T.length()) {
    if (S.length() < T.length()) {
        S += s;
    } else {
        T += t;
    }
}
if (S == T) {
    cout << 1 << '\n';
} else {
    cout << 0 << '\n';
}
```

무한 문자열

https://www.acmicpc.net/problem/12871

- 소스: http://codeplus.codes/9fc0cebd2ab64423959762cd94be9adb

A와 B

https://www.acmicpc.net/problem/12904

- S를 T로 바꾸는 문제
- 가능한 연산
- 문자열의 뒤에 A를 추가한다
- 문자열을 뒤집고 뒤에 B를 추가한다

A와 B

https://www.acmicpc.net/problem/12904

- T의 마지막 문자가 A라면,
- 마지막에 문자열의 뒤에 A를 추가했다는 것이다.
- T의 마지막 문자가 B라면,
- 마지막에 문자열을 뒤집고 뒤에 B를 추가했다는 것이다.

A와 B

https://www.acmicpc.net/problem/12904

- T에서 S를 만들 수 있는지 보는 문제

A와 B

https://www.acmicpc.net/problem/12904

- 소스: http://codeplus.codes/564c14b8cc57445a8791df73c9d1210e

끝

코드 플러스

https://code.plus

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.